MACHINE LEARNING AND DATA ANALYTICS

EE0005 Crash Revision

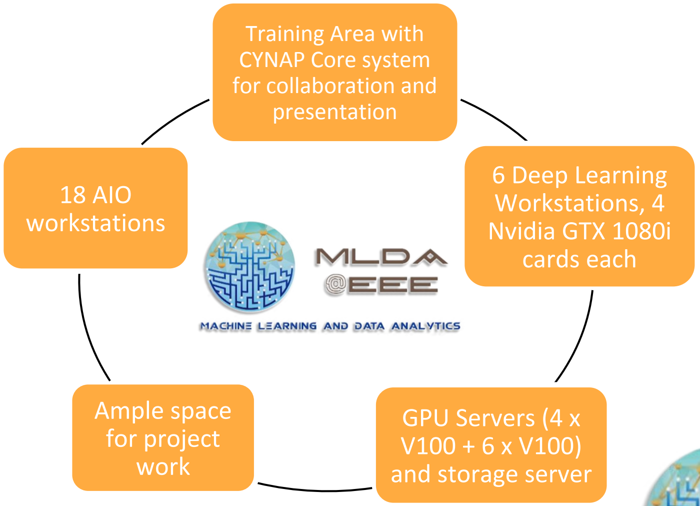
We will begin at 7.10 pm!

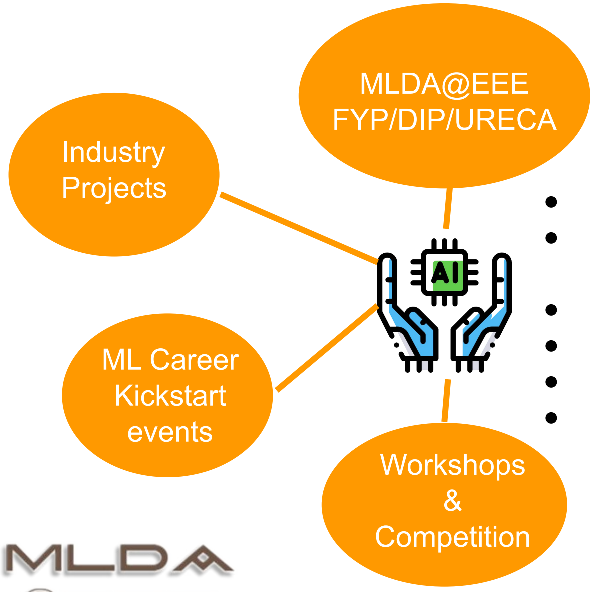
Instructors:

(1) Philip Lee   (2) Lee Jinhen

Our Mission

Consolidate all EEE AI activities under one roof while providing students with an avenue to learn, research and apply ML/DA/AI skills and knowledge to both academia and the industry.





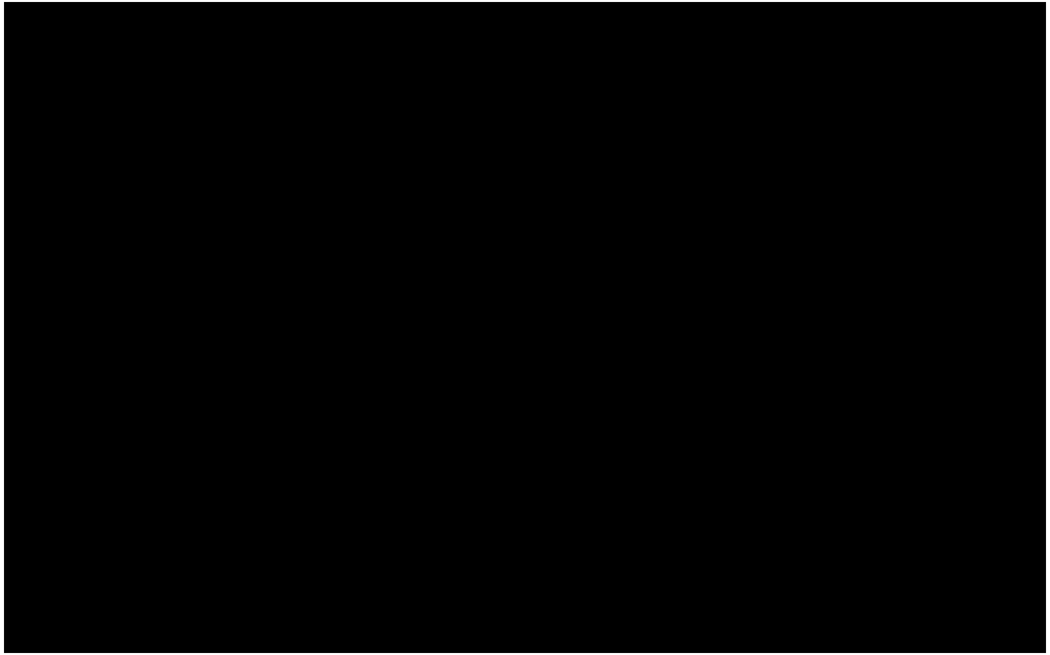
- >17 Workshops
- >1000 Trained ML practitioners
- >10 Academic Projects
- >30 Industry projects
- >5 competition
- >15 Industry Partners

MLDA @EEE
MACHINE LEARNING AND DATA ANALYTICS

Introduction

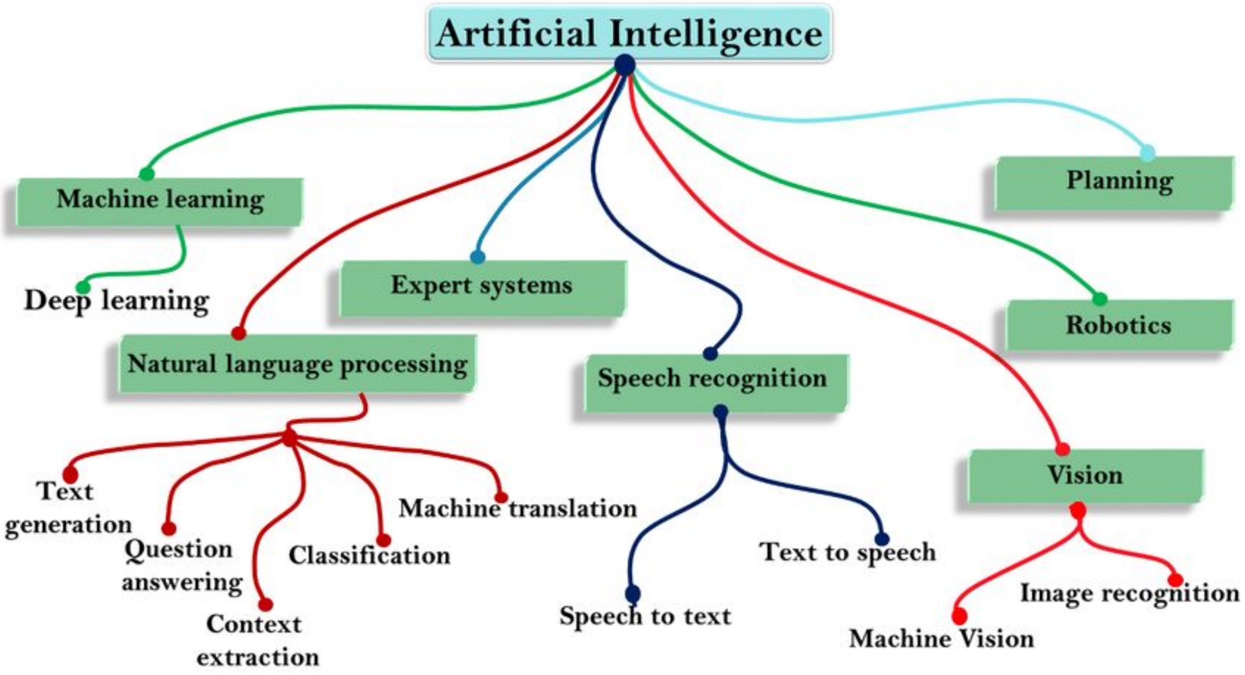
AI: Intelligence demonstrated by machines

OpenAI’s GPT-3: May 2020

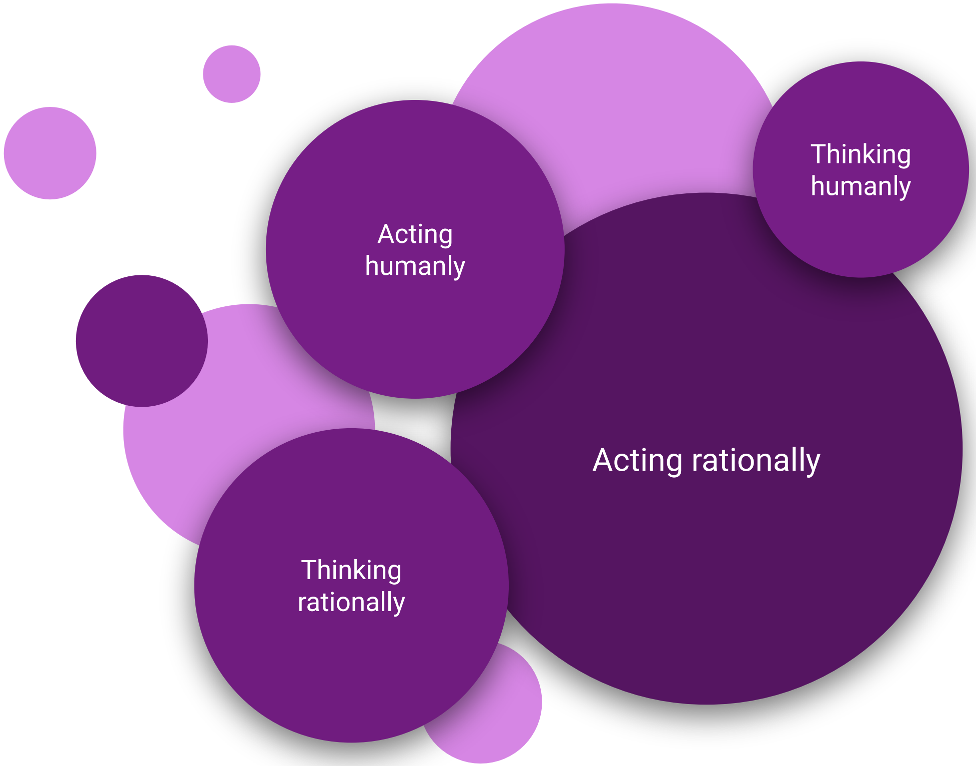


https://nesslabs.com/gpt-3-future-productivity https://twitter.com/sharifshameem/status/1283322990625607681

Major Subfields

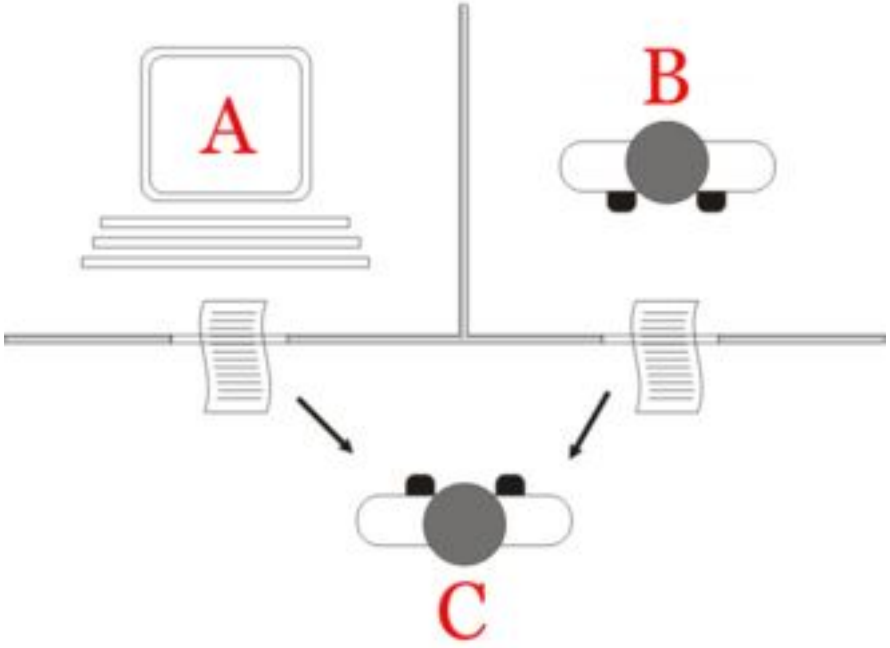


Views on AI

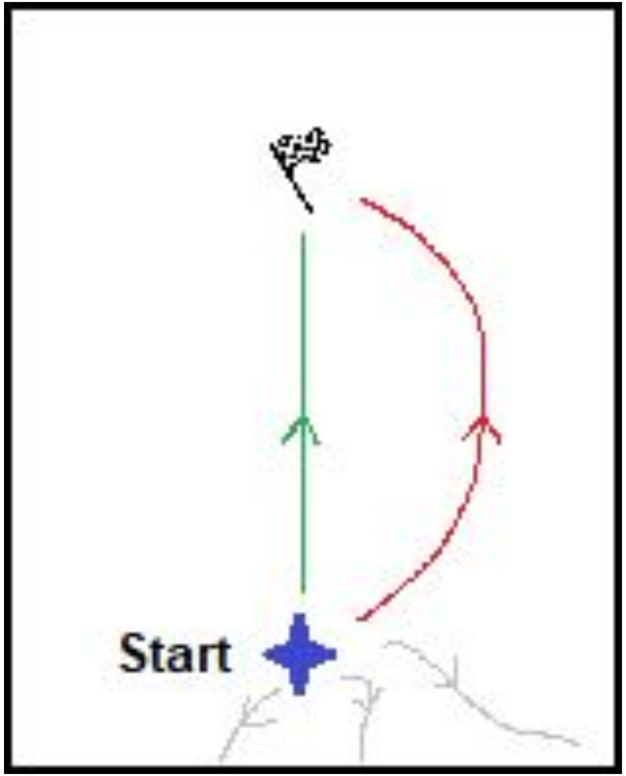

Acting humanly
Thinking humanly
Acting rationally
Thinking rationally

Turing Test

Human interrogates entity via teletype for 5 minutes. If, after 5 minutes, human cannot tell whether entity is human or machine, then the entity must be counted as intelligent

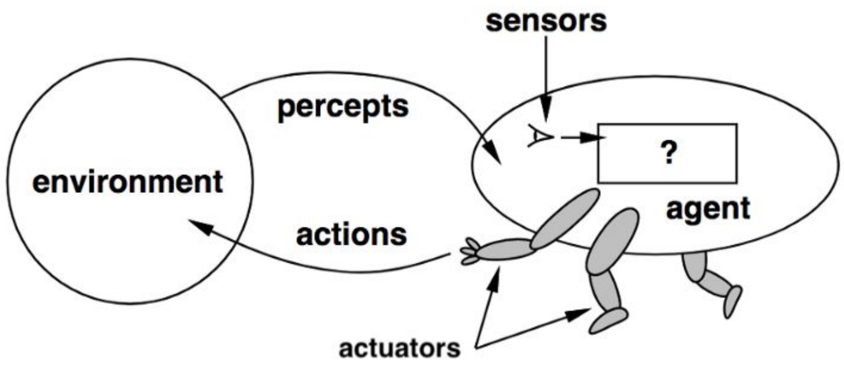


Intelligent Agents



Rational Agents

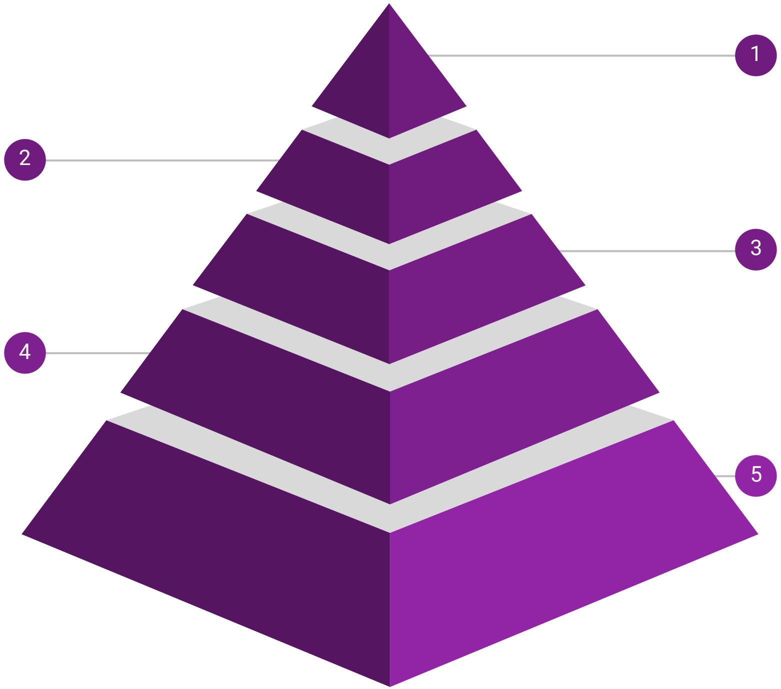
- An agent is an entity that perceives through sensors and acts through effectors



- Rational action:

Expected value of objective performance measure (see later) + Clear preferences (hard-coded) → Choose action with optimal expected outcome

Types of Agents



1 **Simple Reflex Agents**
Agents acts based on rules and perceived data only.

2 **Reflex Agents with State**
Agent acts based on the previous state(s), rules and perceived data.

3 **Goal-based Agents**
Does the action lead the agent to a step closer to the goal?

4 **Utility-based Agents**
Many actions lead to the same goal. What is the utility of a certain state?

5 **Autonomous Agents**
Adapts to the environment through experience.

Types of Environment

(Example: Minesweeper)

01	**Accessible / Inaccessible**	• Does the agent know the complete environment state? ○ *Inaccessible. Mines are hidden so imperfect information.*
02	**Deterministic / Nondeterministic**	• Can the next state be completely determined by the current state and action taken? ○ *Nondeterministic. Mines are randomly placed in diff. positions.*
03	**Episodic / Sequential**	• Does the agent require the memory of past actions to determine the next best actions, or only current percepts are important? ○ *Sequential. The actions are based on previous outcomes.*
04	**Static / Dynamic**	• Does the environment change while the agent deliberates? ○ *Static. When are you considering the next step, no changes in environment.*
05	**Discrete / Continuous**	• Does the environment have a discrete actions or states? ○ *Discrete. All positions and movements are in discrete domains.*

Pop Quiz

What kind of observing environments are present in artificial intelligence?

A. Partial
B. Fully
C. Learning
D. Both Partial & Fully

Design of a Problem-Solving Agent

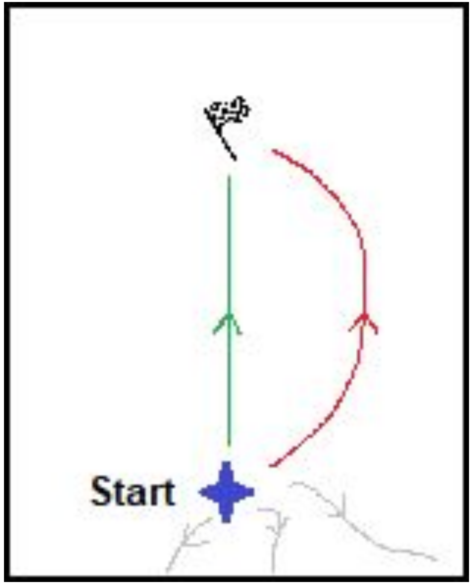
Consider the expected outcomes of different possible sequences of actions that lead to states of known value.

Steps:

1. Goal formulation eg. minimise Time / Distance
2. Problem formulation
3. Search process
   - No knowledge → uninformed search
   - Knowledge → informed search
4. Action execution (follow the recommended route)

Problem Formulation

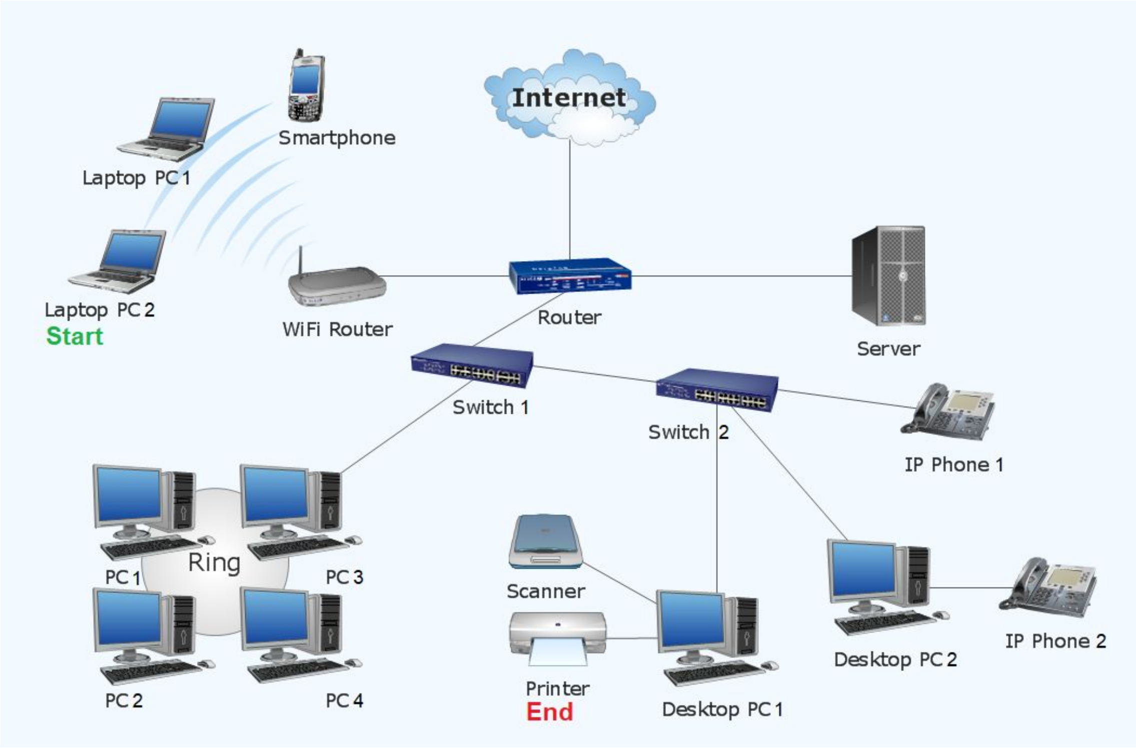
- Initial state - eg. starting position
- Actions set - eg. up, down, blink
- State space - all the states reachable, may be infinite
- Goal test - what is the goal, may have >1 states
- Cost function = Search cost ("offline")
  + Execution cost ("online")
  - Trade-off is required, a "good enough" solution can be found in a shorter time
- Solution can be a path / a sequence of actions



Try Yourselves!

(Example: Computer Networking)

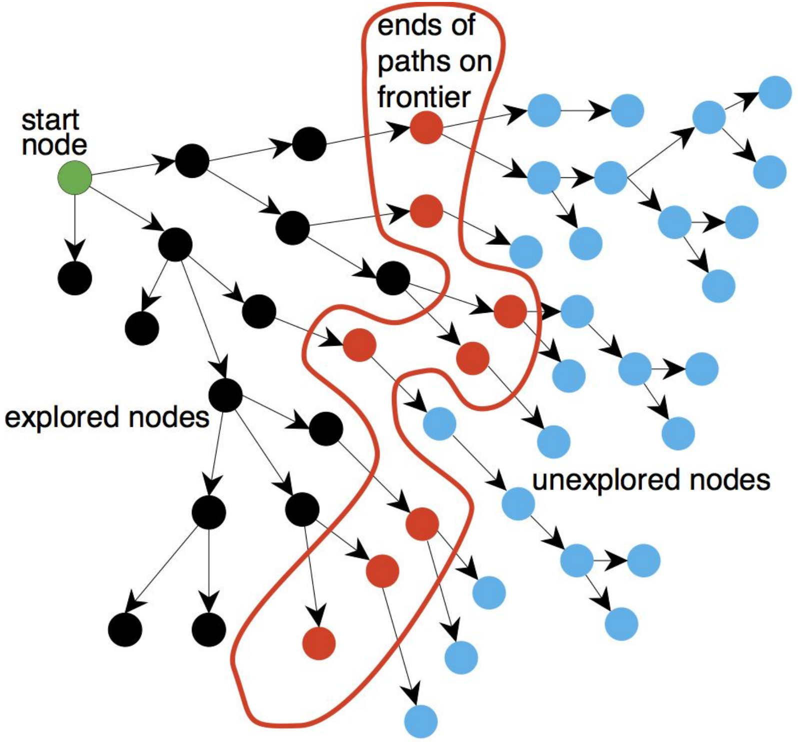
- Initial state
- Actions set
- State space
- Goal test
- Cost function



Search

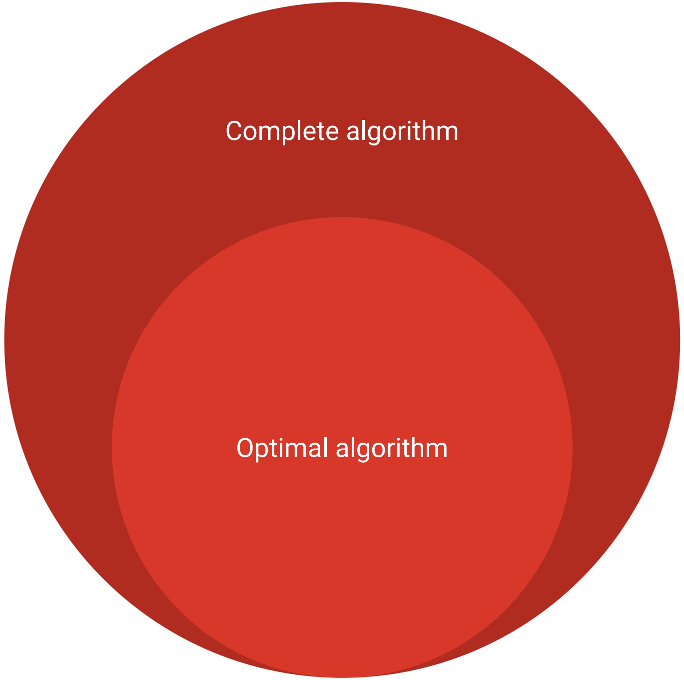
Search algorithm

- Traversing the state space along a "frontier"
  - Frontier: Foremost unexpanded nodes
- Explored nodes are stored in a "explored set" so that we don't expand it again



Search strategy

- We need a search strategy to determine which node to expand
- Classified according to:
  - Termination: Guaranteed to terminate no matter how large the state space
  - Completeness: Guaranteed to find a solution (if there is one)
  - Time complexity: Total number of nodes expanded/generated
  - Space complexity: Maximum number of nodes in memory at any one time
  - Optimality: Guaranteed to find the best (?) solution (if there is one)



Search strategy (cont.)

- Uninformed search strategies (Blind search)
    - Use only problem definition: no information about goal state at all
        - Breadth-first search
            - Uniform-cost search
        - Depth-first search
            - Depth-limited search
                - Iterative deepening search
- Informed search strategies
    - Use problem-specific knowledge: information about goal state, “forward-looking” knowledge
        - Best-first search
            - Greedy search
            - A* search

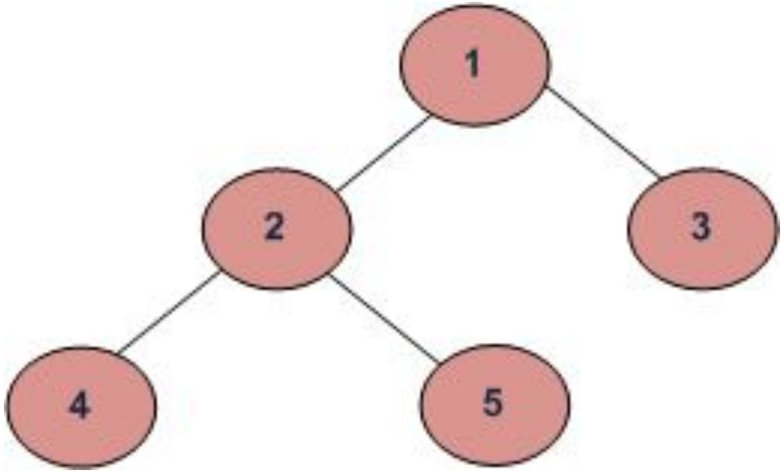
Breadth-first search/Uniform-cost search

- Implemented with FIFO queue
    - Uniform-cost search: With priority queue

1					

1	2	3			

2	3	4	5		



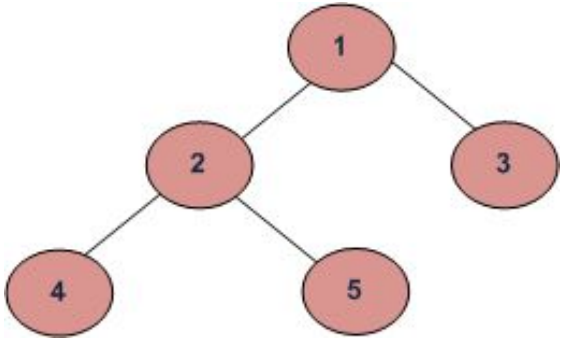
Breadth-first search/Uniform-cost search

- Time complexity:
  - Expand every node, denote maximum branching factor b and solution path length d

$$1 + b_1 + b_1 b_2 + \cdots + b_1 b_2 \ldots b_d \leq 1 + b + b^2 + \cdots + b^d = O(b^d)$$

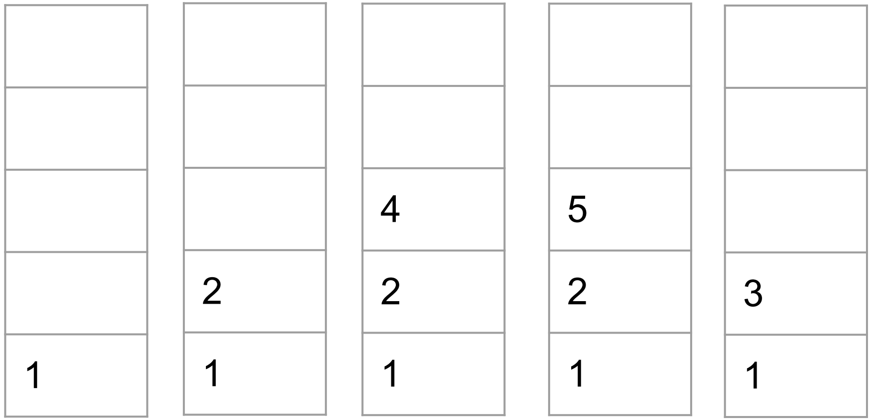
- Space complexity:
  - Solution is at the last node of the deepest level (depth d)

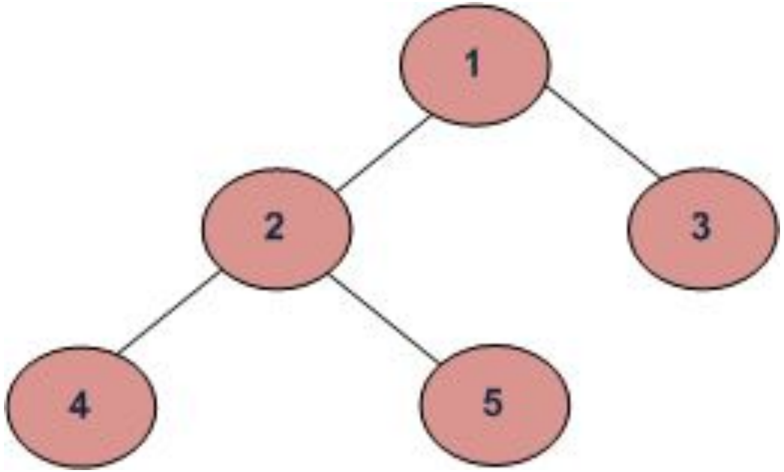
$$b_1 b_2 \ldots b_d \leq b^d = O(b^d)$$



Depth-first search

- Implemented with LIFO stack
    - Depth-limited search
        - Iterative deepening search





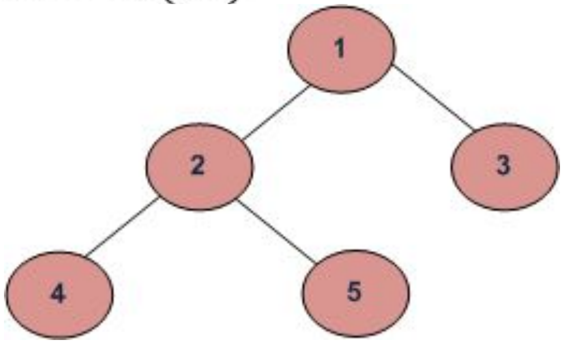
Depth-first search

- Time complexity:
  - Expand every node, denote maximum branching factor b and maximum depth d

$$1 + b_1 + b_1 b_2 + \cdots + b_1 b_2 \ldots b_d \leq 1 + b + b^2 + \cdots + b^d = O(b^d)$$

- Space complexity:
  - For each node, need to store siblings (same level) to use when backtracking

$$1 + b_1 + b_2 + \cdots + b_{d-1} \leq b + b + b + \cdots + b = bd = O(bd)$$



Pop Quiz

Consider an infinite-depth search tree with small branching factor, and the goal is located at a “shallow” level. BFS is preferred over DFS in this case, why?

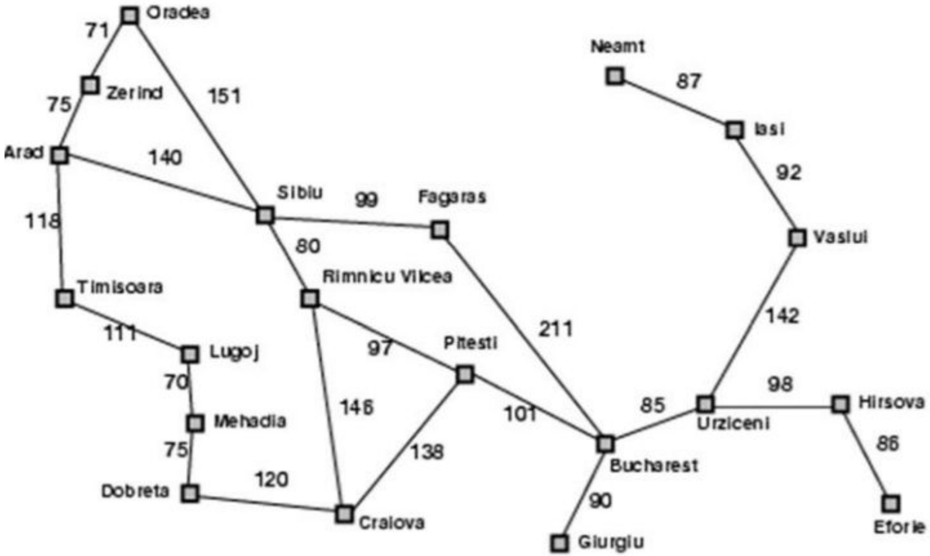
A. DFS is incomplete in this case.
B. BFS is much more efficient to “prune” the search tree.
C. DFS focuses on reaching the leaf of a particular path.
D. All of the above.

Evaluation function

- Path cost function g(n)
    - Backward looking
    - From initial state to current state
    - Used in uniform-cost search
- Heuristic function h(n)
    - Forward looking
    - From current state to goal state
    - <u>Estimated</u> for the cheapest path!
    - Admissible: a function that does not overestimate <u>real</u> cost of reaching the goal

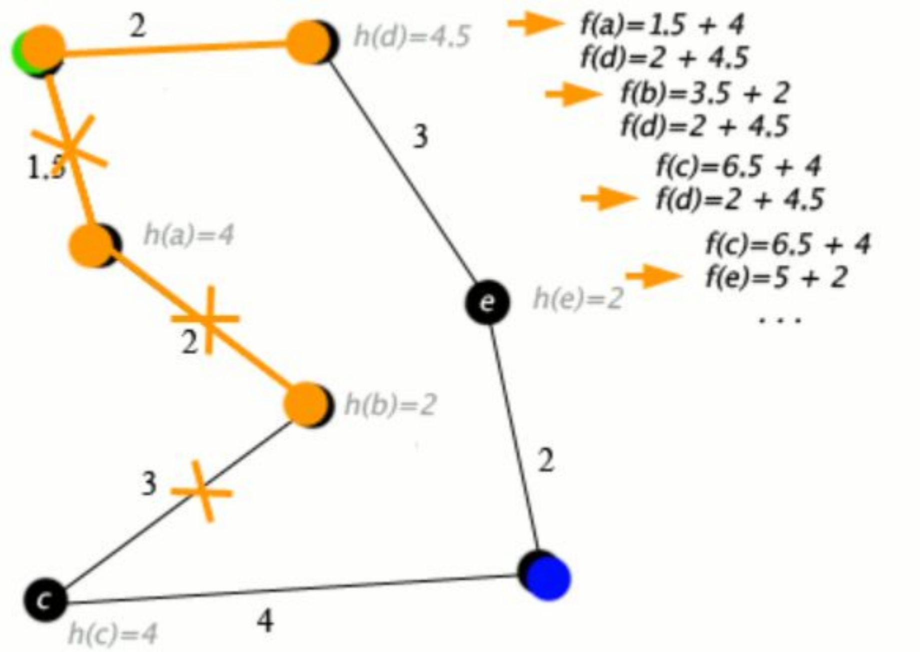
Greedy search

- Greedy: Finds solution in an incremental manner based on local optimality
  - Expands the node which appears closest to goal
  - Not complete, why?
- Heuristic function, h(n)
  - Estimate of the distance from current state to goal state
  - At the goal, h(goal) = 0



A* search

- Uniform-cost search (g(n)) + Greedy search (h(n))
  - Total cost from start to goal, passing through node n
  - Evaluation function: g(n) + h(n)
- Disadvantage: exponential space complexity
  - Store every expanded node in memory



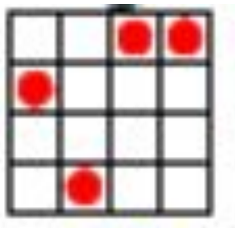
Constraint Satisfaction & Game Playing

Constraint Satisfaction Problem

Goal: Discover some state that satisfies the constraints set

State: A combination of some/all variables' arrangement with values in the domains

- Eg. Variables (eg. Timetable slots), Domains (eg. Time in a working day)
- An assignment that does not violate any constraints is called a *consistent* or *legal* assignment.
- A solution to a CSP
  - Every variable given a value (*complete*)
  - The assignment satisfies all the constraints

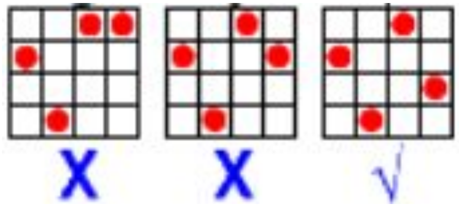


Goal test: A set of constraints

- Eg. Timetable slots cannot clash → Useful for timetable scheduling

Imagine this scenario - 4 Queens

4x4 grid. No two queens on the same row, column or diagonal

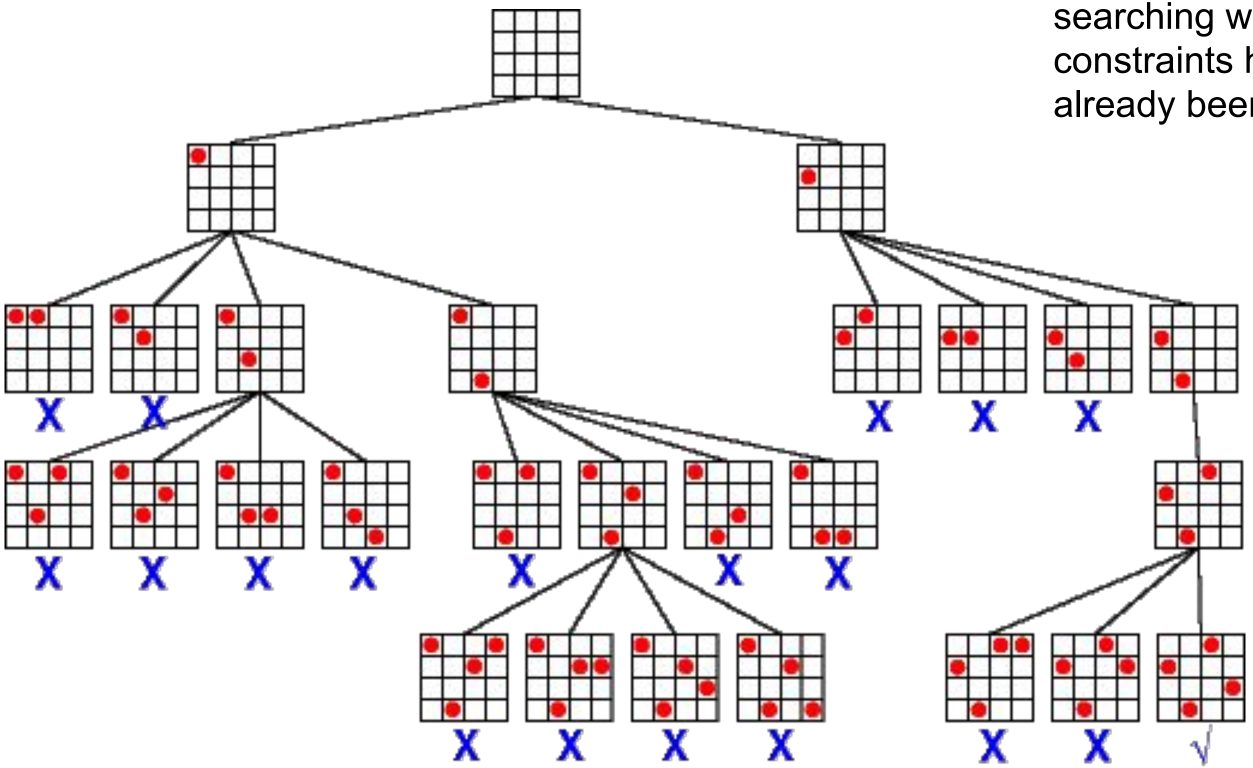


We search column by column

Method 1: List all possibilities, then check all

- 4^4 = 64 outcomes
- Time complexity: $O(n^n)$
- Depth-first standard search

Backtracking search



Do not waste time searching when constraints have already been violated!

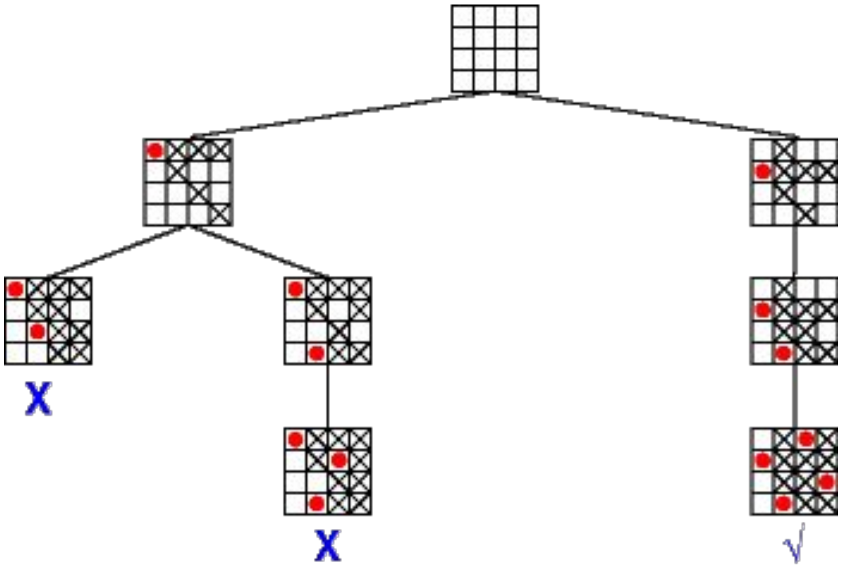
Forward tracking and constraint propagation

More ‘intelligent’ search

- Implications of current variable assignments for the other unassigned variables
- Which variable to assign next
- What order of the values to try for each variable

Note:

1. Keep track of remaining legal values for unassigned variables
2. Terminate search when any variable has no legal values



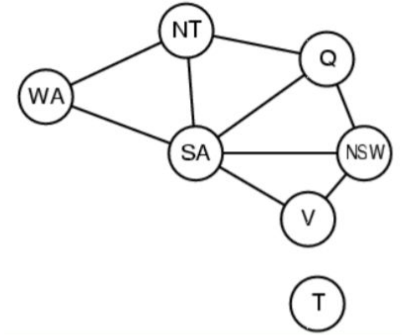
How to reduce search size?

More ‘intelligent’ search - 1 important method, 3 heuristics

Wikipedia:

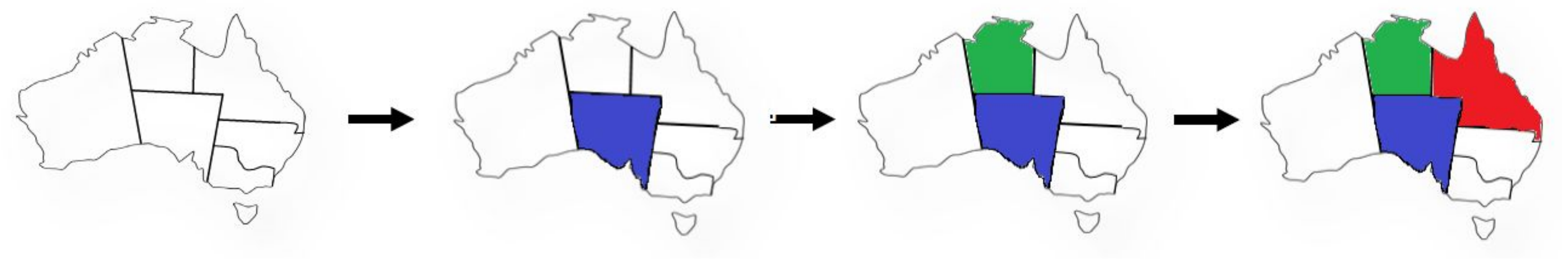
A heuristic technique, or a **heuristic**, is any approach to problem solving or self-discovery that employs a practical method that is not guaranteed to be optimal, perfect, or rational, but is nevertheless sufficient for reaching an immediate, short-term goal or approximation.

Which variable to fill first?



- Which variable to assign next
- What order of the values to try for each variable

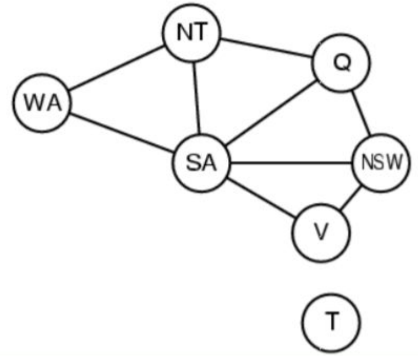
Most Constrained Variable (aka minimum remaining values heuristic)

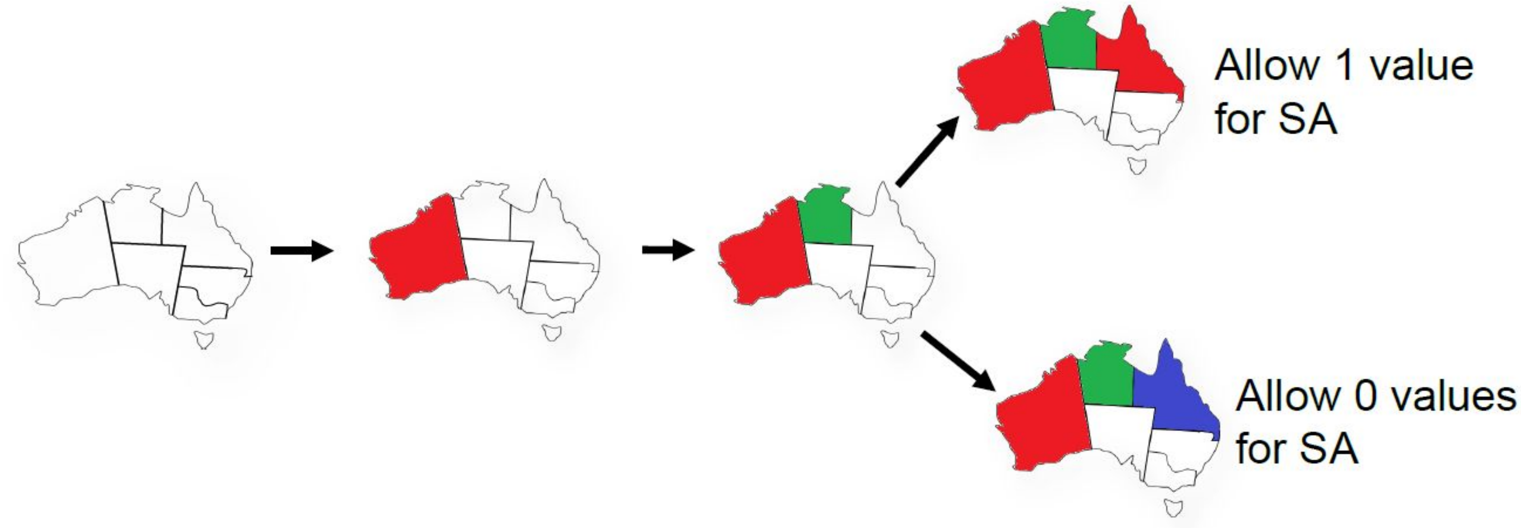


- Choose the variable with the fewest legal values
- Reduces the branching factor on future choices!

Least Constraining Value Heuristic

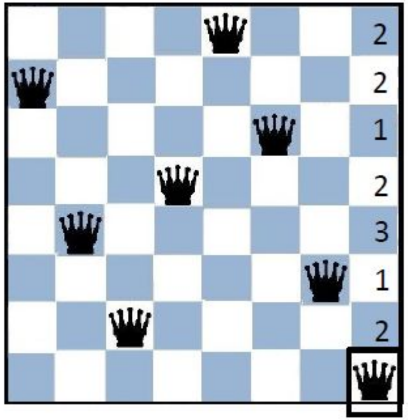
Choose the variable that rules out the fewest values / allows maximum flexibility in the remaining variables.



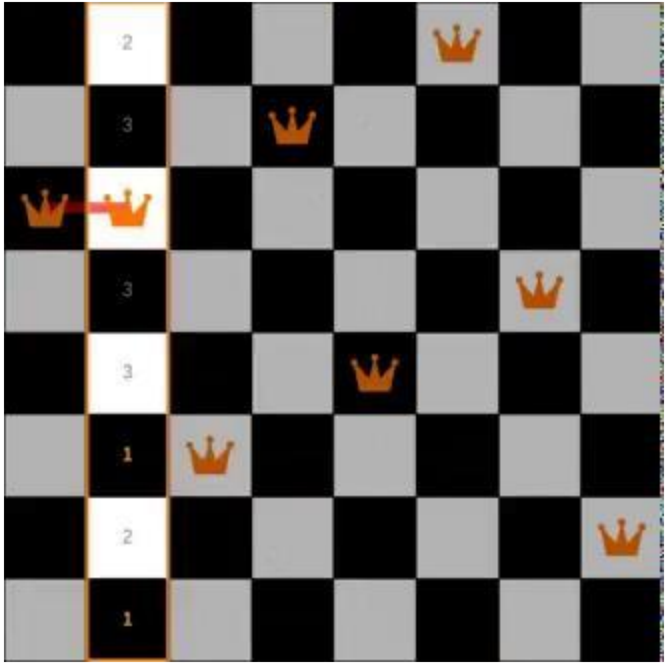


Min-Conflicts Heuristic

- Assign a initial value for each variable
  - Greedy algorithm for this example
- Shifts violated variable to a domain with min. no. of violated constraints



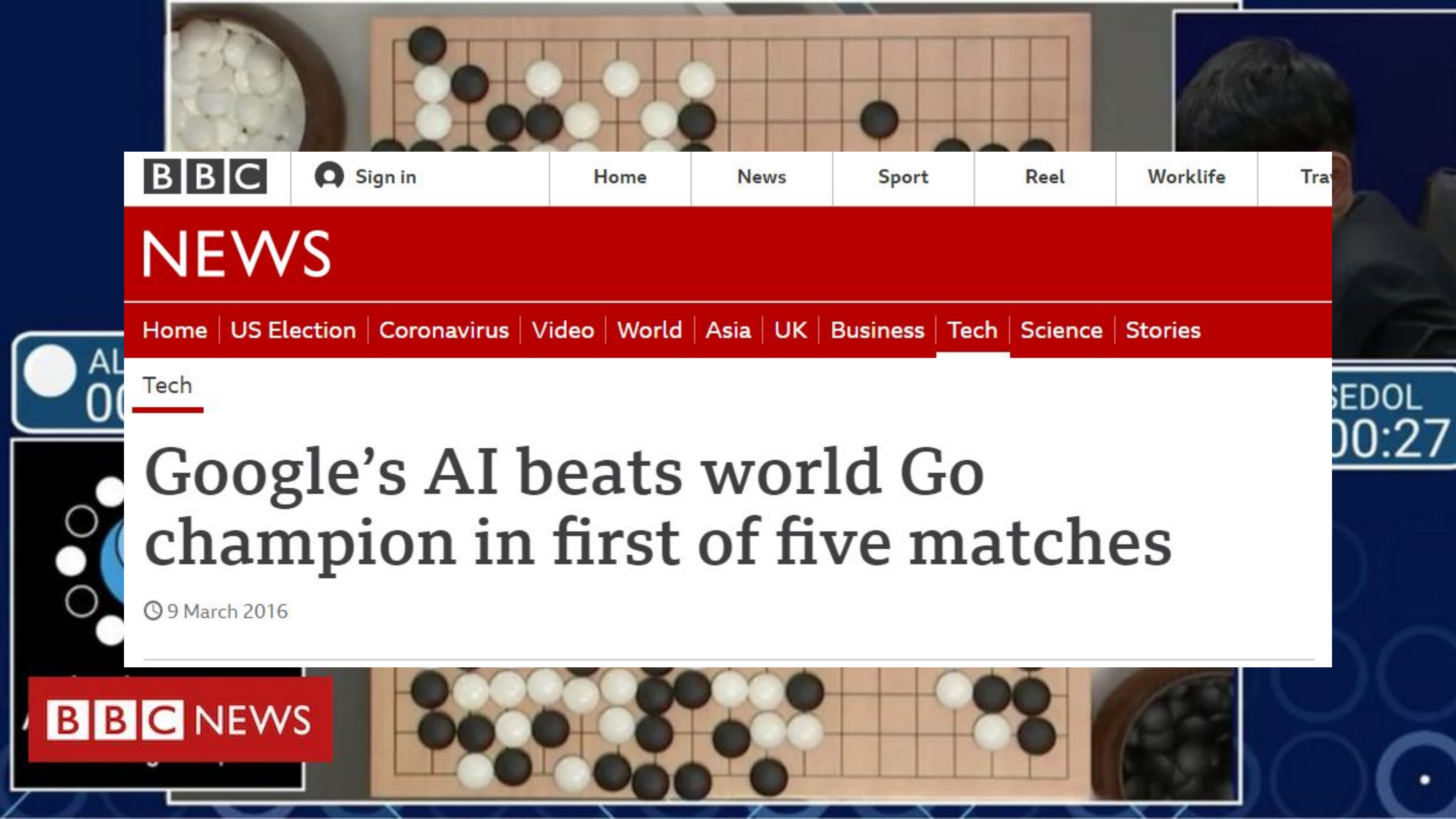
- Might not reach end goal, local search



Games as Search Problem

Games are commonly used as a benchmark in the AI industry.

- Abstraction: Ideal representation of real world problems
- Uncertainties: Contingency Problem
  - The agent doesn't know what effect its actions will have. This could be due to the environment being partially observable, or because of another agent.
- Complex: Simple rules but huge branching factor
  (Real world problems might have no end state)
- Time-limited: Trade-off is required; A “good enough” solution is not optimal
  Search efficiency is crucial



BBC | Sign in | Home | News | Sport | Reel | Worklife

NEWS

Home | US Election | Coronavirus | Video | World | Asia | UK | Business | Tech | Science | Stories

Tech

Google's AI beats world Go champion in first of five matches

9 March 2016

Pop Quiz

Which type of Game/environment is Chess?

A. Perfect information, Episodic & Dynamic
B. Perfect information, Sequential & Static
C. Imperfect information, Episodic & Static
D. Imperfect information, Sequential & Dynamic

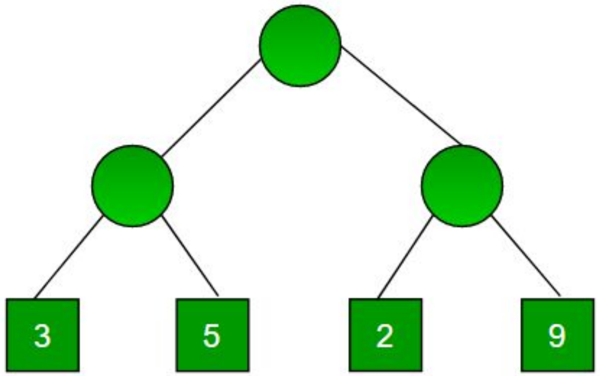
Minimax Search Strategy

Maximise one's own utility and minimise the opponent's utility

- Assumption is that the opponent does the same → minimise your utility

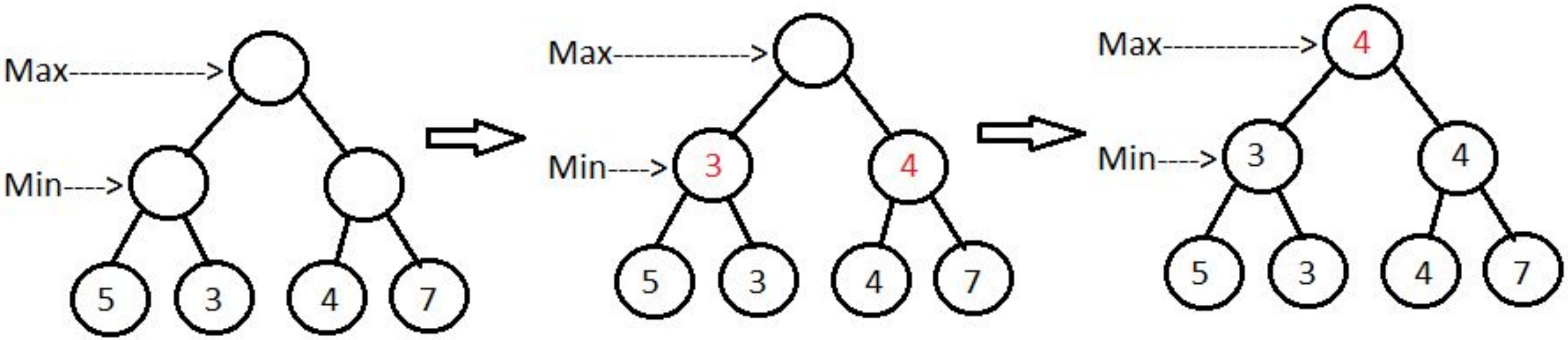
Perfect decisions: no time limit is imposed

1. Generate the complete game tree from initial to terminal states
2. Calculate utility of terminal state and back propagate to parent of terminal state. Repeat the process till initial state is reached.
3. Select the move with the highest utility value.



* Note: Utility function - a numeric score to quantify the outcome of a game

Minimax Search Strategy - Perfect Decision



Imperfect Decisions

Previous search strategy is intractable for games with a large branching factor!

Modify the minimax algorithm

- Replace utility function with an evaluation function
- Only do partial tree search (eg. depth of 2 or 3)

Evaluation function? There are numerous heuristics. Examples are:

- Domain experts can advice on the probability of each state winning
- Use of neural network

Pop Quiz

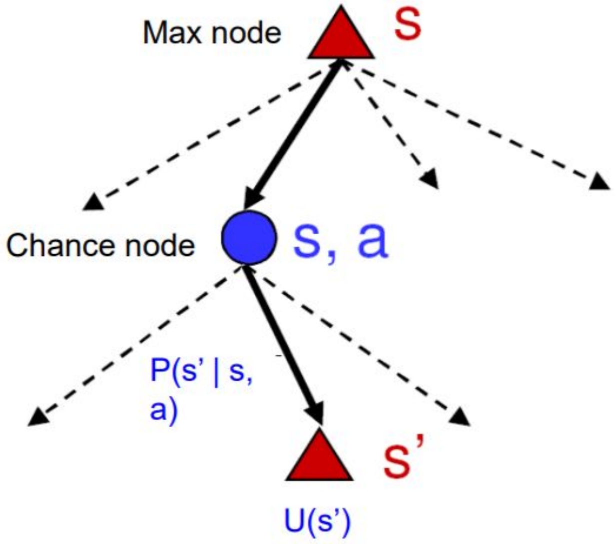
Which function is involved when the agent is deciding which choice to make and have no time limit imposed?

A. Utility function
B. Propagation function
C. Evaluation function
D. Minimax function

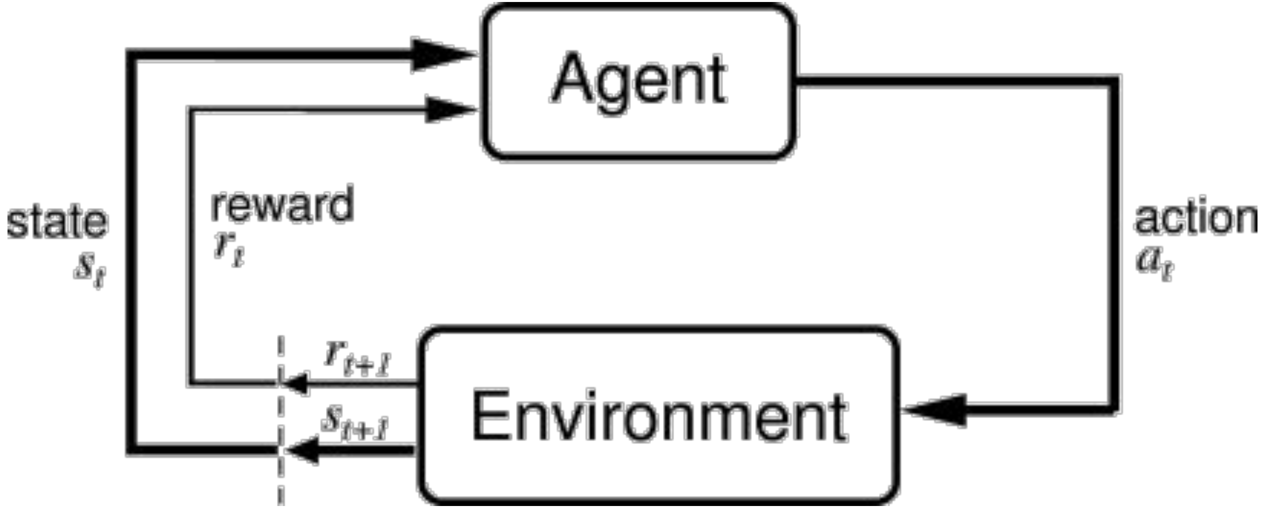
Agent Decision Making & Reinforcement Learning

Utility theory

- Expected utility:
  - $$EU(A \mid E) = \sum_i P(Result_i(A) \mid E, Do(A))U(Result_i(A))$$
  - Principle of maximum expected utility: Agent should choose action A under environment E that gives highest expected utility



Agent-environment interface



Markov Decision Process (MDP)

- Sequential decision making
- Transition properties depend <u>only</u> on the current state (Markov Property)

State space *S*	Describes the possible "condition" of the agent
Action space *A(s)*	A set of actions *a* that the agent can take at state *s*
Transition model *P(s' | s, a)*	Probability that the agent will reach state *s'* from current state *s* by taking action *a*
Reward function *R(s)*	"Reward" for reaching state *s*. Total of these rewards to be maximised
Policy π*(s)*	The <u>solution</u>: optimal action to be taken at state *s*

Utilities of state sequences

- Goal: To have an optimal policy maximising expected utility over <u>all</u> state **sequences** ("true" utility of a **state**)
  - Problem: infinite state sequences = unbounded total utility
  - Solution: Recent reward counts more than previous rewards, <u>discount factor γ</u>

$$U([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$$

- Optimal policy

$$\pi^*(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$$

Bellman optimality equation

$$U([s_0, s_1, s_2, \dots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots$$

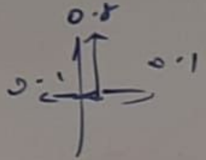
$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$$

- Value iteration
  - *U* = 0 for all states *s*

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U_i(s')$$

- Policy iteration
  - Start with initial policy π_0
  - Calculate utility function using current policy

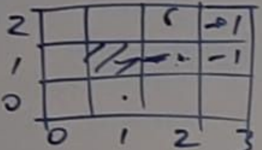
$$\pi^{i+1}(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s' | s, a) U^{\pi_i}(s')$$



$U_0(s) = 0$

$\forall s \in S$

Value iteration example



The two cells have rewards as shown. The other cells have reward -0.04

Assume a discount factor of 0.9 and treating the cell crossed out as an obstacle (illegal state), use value iteration to find the optimal utility.

$-0.04 + 0.9 \times (0.8 \times 0 + 0.1 \times 0 \ldots)$ [illegible]

$= \boxed{-0.04}$

$U_1(s) \leftarrow -0.04$

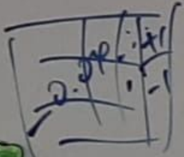
(2, 2)

$U_1((2,3)) = 1$

$U_1((1,3)) = -1 + 0.9 \ldots = -1$

$0.778 = 0.6728$

$$U_{i+1}(s) = R(s) + \gamma \boxed{\max_{a \in A(s)} \sum_{s'} P(s'|s,a)\, U_i(s')}$$



up: 0.064

right: 0.782

left: −0.04

down: −0.132

$0.8\, U_1((0,2)) + 0.1\, U_1((1,2)) + 0.1\, U_1((1,3))$

$= -0.8 \times 0.04 + 0.1 \times -1$

$= -0.132$

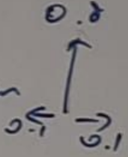
Actions space

$A(s) = \{\text{up, down, left, right}\}$.

Transitions model.

$P(s'|s,a)$

Decides to go a given direction, there is 0.8 probability going in that direction, 0.1 veering left & 0.1 veering right. If hitting an obstacle, it will remain in original position.



① Initialise all $U_0(s) = 0$

② iteration 1

$U_1((2,2)) = -0.04 + 0.9(0.8 \times 0 + 0.1 \times 0 + 0.1 \times 0) = -0.04$

$U_1((1,2)) = -0.04 + 0.9(0.8 \times 0 + 0.1 \times 0 + 0.1 \times 0) = -0.04$

$U_1((2,3)) = +1 + 0.9(0.8 \times 0 + 0.1 \times 0 + 0.1 \times 0) = 1$

$U_1((1,3)) = -1 + 0.9(0.8 \times 0 + 0.1 \times 0 + 0.1 \times 0) = -1$

③ iterations 2

$U_2((2,2)) = -0.04 + 0.9(0.8 \times 1 + 0.1 \times -0.04 + 0.1 \times -0.04) = 0.6728$

$U_2((2,3)) = 1 + 0.9(0.8 \times 1 + 0.1 \times 1 + 0.1 \times -0.04) = 1.8064$

Temporal difference (TD) prediction

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right]$$

- Does not require a model of the environment (e.g. transition model)
- Fully incremental
  - Learn <u>before</u> knowing final outcome
  - Learn <u>without</u> knowing final outcome

Pop Quiz

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right]$$

Refer to the iteration scheme above. Does the value of α affect the rate of convergence of the state value function? (Hint: Think gradient descent)

A. No, α depends only on the number of possible state.
B. Yes, it dictates how large is the correction step is.
C. No, the rate of convergence depends solely on γ.
D. Yes, the value of α must be larger than 1 to guarantee convergence.

Q & A

Thank you

We appreciate some feedback

https://forms.gle/EgnErxgjW743Jz1w6

SCAN ME

Photo-taking